HW07 Application exercises: Egalitarianism and income

Ellen Hsieh

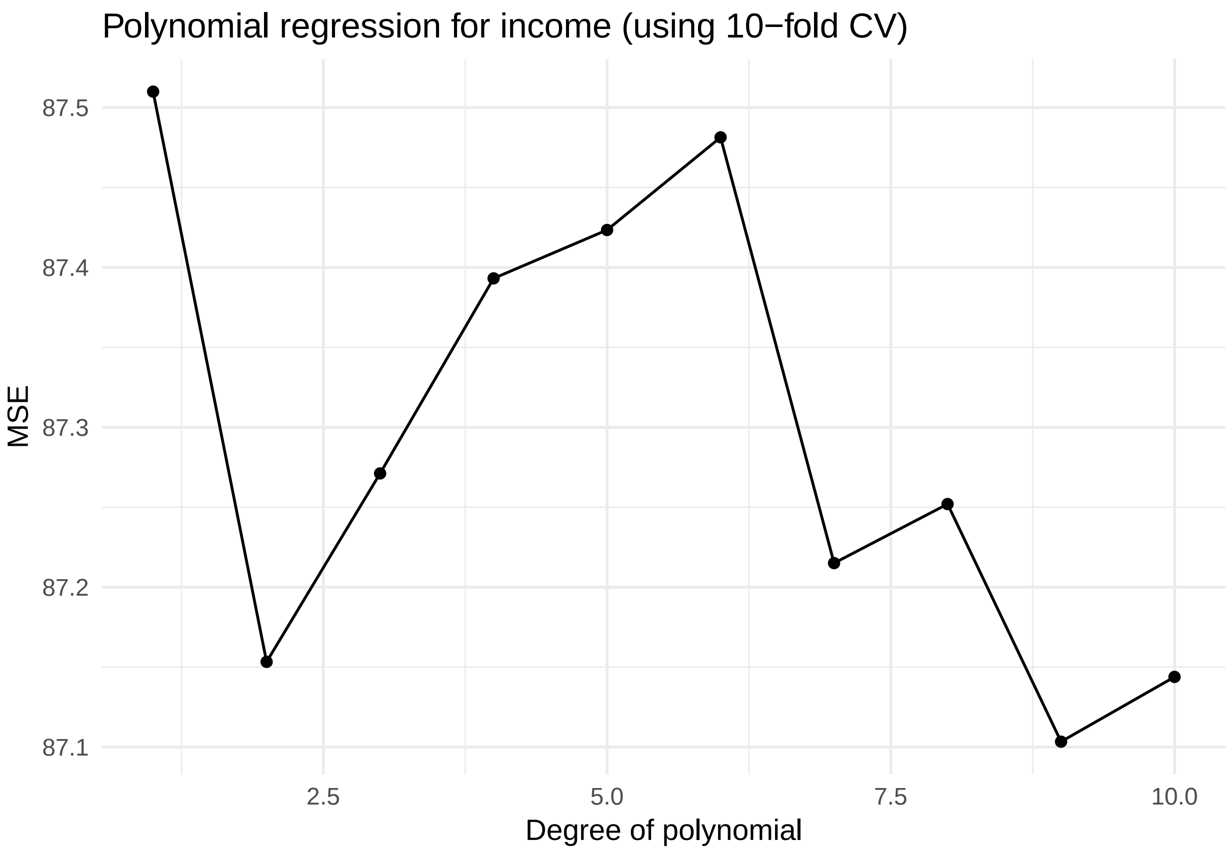
```r
library(tidyverse)
library(ggplot2)
library(rsample)
library(broom)
library(rcfss)
library(splines)
library(margins)

theme_set(theme_minimal())
set.seed(124)
```

```r
# load the data
gss_train <- read_csv("./data/gss_train.csv")
gss_test <- read_csv("./data/gss_test.csv")
```

1. Polynomial regression

```r
# get the train income data
inc_train <- gss_train %>%
  select(egalit_scale, income06)

# get 10-fold cross-validation of training income data
inc_train_cv <- vfold_cv(data = inc_train, v = 10)
poly_mse <- vector(length = 10)

# calculate the MSE of polynomial regression
for (i in 1:10){
      splited_data <- inc_train_cv$splits[[i]]
      train_data <- analysis(splited_data)
      holdout <- assessment(splited_data)
      actual <- holdout$egalit_scale
      for (d in 1:10){
            mod <- glm(egalit_scale~ poly(income06, d, raw = T), data = train_data)
            pred <- predict(mod, newdata = holdout)
            mse_temp <- sum((pred - actual)**2) / length(pred)
            poly_mse[d] <- poly_mse[d] + mse_temp
      }
}

# get the average MSE of polynominal regression
poly_mse <- poly_mse / 10
p_tibble <- tibble(MSE = poly_mse, Degree = 1:10)
min_poly_mse <- which.min(p_tibble$MSE)

# plot the MSE if polynomial regression
p_tibble %>%
  ggplot(aes(x = Degree, y = MSE)) + geom_point() + geom_line() +
  labs(title = "Polynomial regression for income (using 10-fold CV)",
```

```
       x = "Degree of polynomial",
       y = "MSE")
```



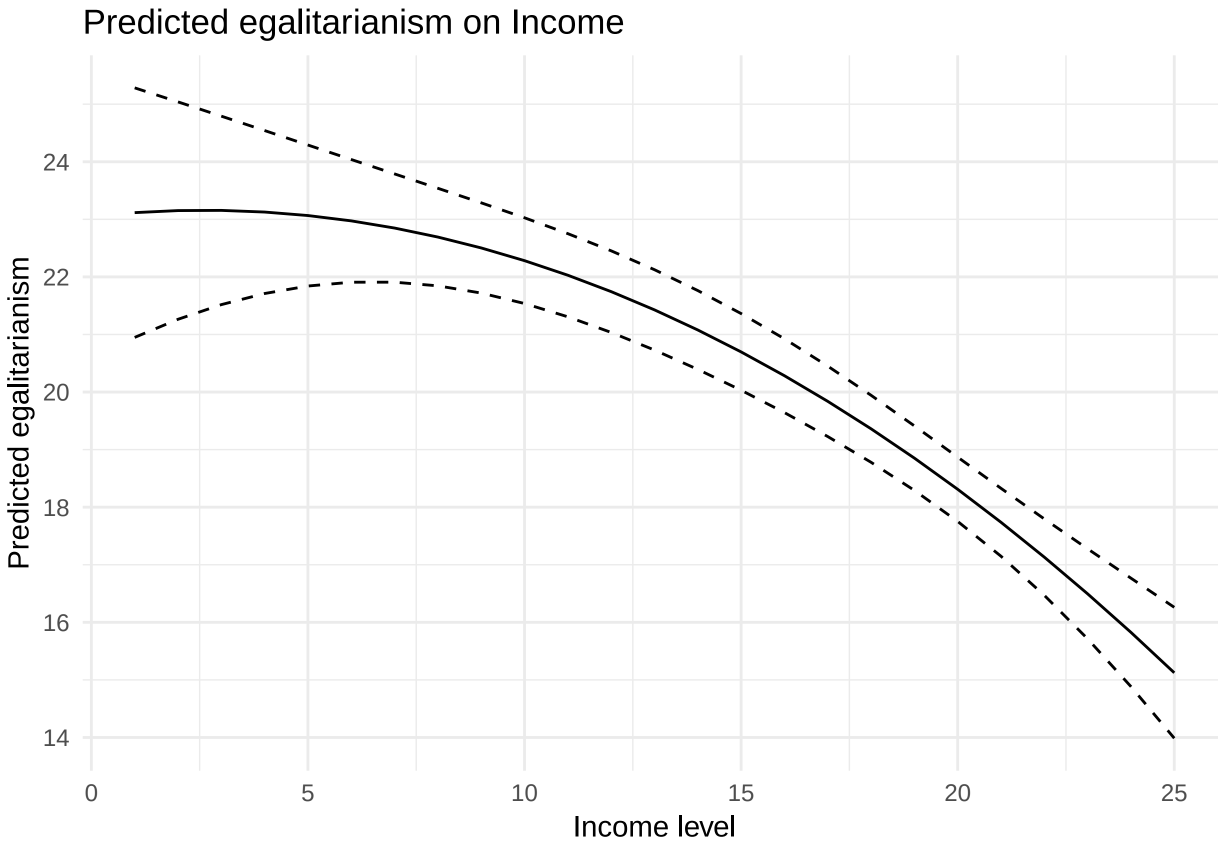
The optimal degree is 2.

```
# fit the optimal polynominal regression model (degree = 2)
poly_res <- glm(egalit_scale ~ income06 + I(income06**2),
                 data = inc_train)
tidy(poly_res)
```

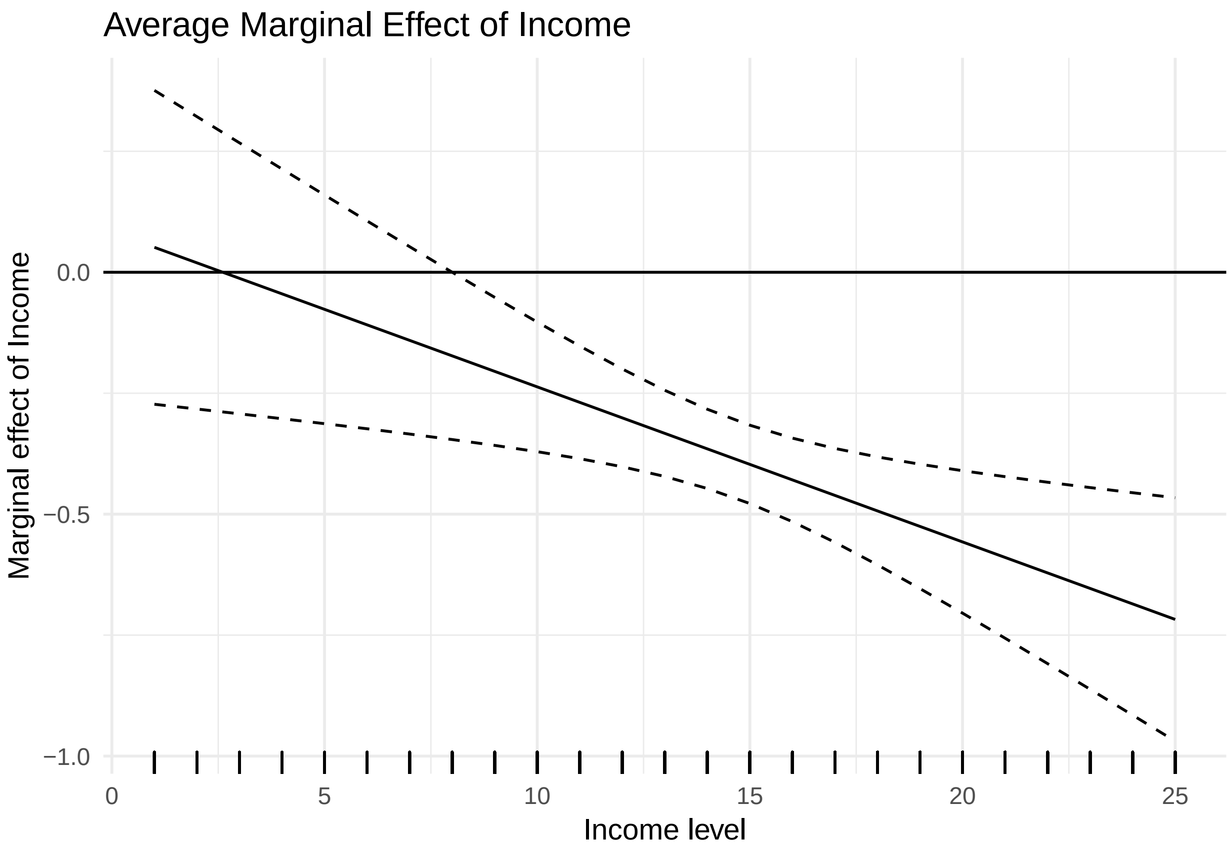
```
## # A tibble: 3 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    23.0      1.26        18.3   2.39e-67
## 2 income06        0.0836   0.177        0.472 6.37e- 1
## 3 I(income06^2)  -0.0160   0.00587     -2.73  6.44e- 3
```

```
poly_res_pred <- augment(poly_res, newdata = inc_train) %>%
  mutate(pred_low = .fitted - 1.96 * .se.fit,
         pred_high = .fitted + 1.96 * .se.fit)

# plot the predicted egalitarianism on income06
ggplot(poly_res_pred, aes(income06)) +
  geom_line(aes(y = .fitted)) +
  geom_line(aes(y = pred_low), linetype = 2) +
  geom_line(aes(y = pred_high), linetype = 2) +
  labs(title = "Predicted egalitarianism on Income",
       x = "Income level",
```

```
       y = "Predicted egalitarianism")
```



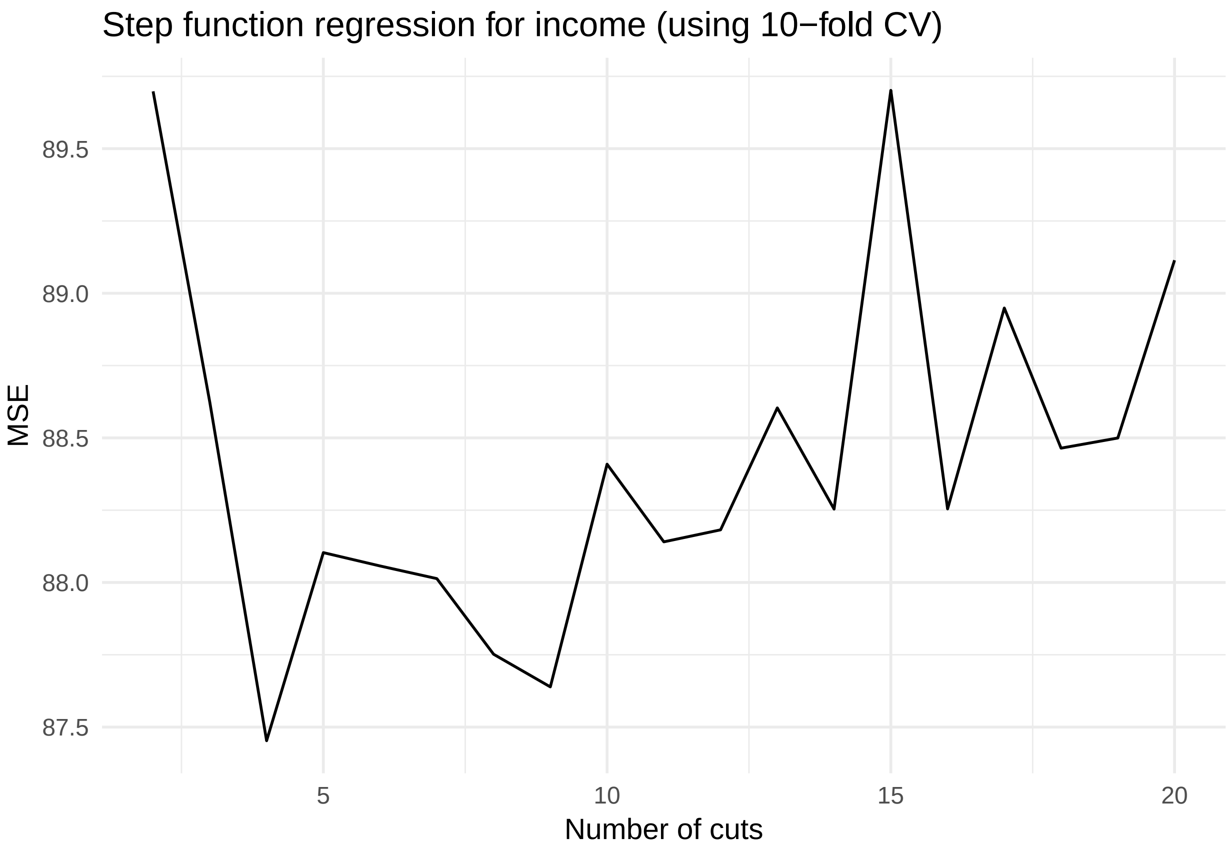

```
# plot average marginal effect (AME)
cplot(poly_res, "income06", dx = "income06", what = "effect", draw = FALSE) %>%
  ggplot(aes(x = xvals)) +
  geom_line(aes(y = yvals)) +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  geom_hline(yintercept = 0, linetype = 1) +
  geom_rug(data = inc_train, aes(x = income06)) +
  labs(title = "Average Marginal Effect of Income",
         x = "Income level",
         y = "Marginal effect of Income")
```



The prediction curve of polynominal regression with degree = 2 is close to a quadratic polynominal. The marginal effect of `income06` on `egalit_scale` decreases as `income06` increases.

2. Step function regression

```
# fit a step function using 10-fold CV
cv.error <- vector(mode = "numeric", length = 19)

for (i in 2:20) {
  gss_train$income06_cut <- cut_interval(gss_train$income06, i)
  glm.fit <- glm(egalit_scale ~ income06_cut, data = gss_train)
  cv.error[i - 1] <- boot::cv.glm(gss_train, glm.fit, K = 10)$delta[1]
}

# plot the MSE for step function
tibble(n_cut = 2:20, mse = cv.error) %>%
ggplot(aes(n_cut, mse)) +
geom_line() +
labs(title = "Step function regression for income (using 10-fold CV)",
         x = "Number of cuts",
         y = "MSE")
```

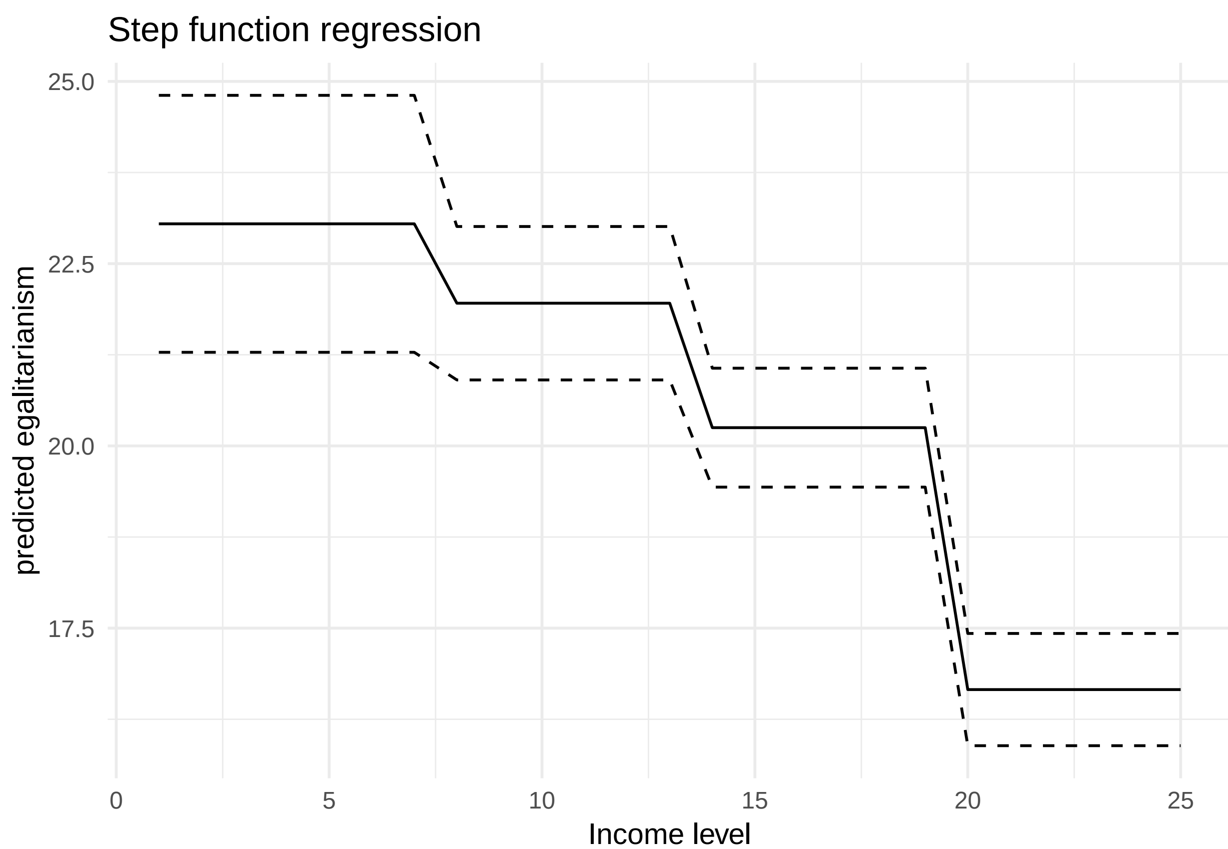


The optimal number of cut is 4.

```
# fit the optimal step function
inc_train$interval_4 <- cut_interval(inc_train$income06, 4)
optimal_step <- glm(egalit_scale ~ interval_4,
                 data = inc_train)
tidy(optimal_step)
```

```
## # A tibble: 4 x 5
##   term              estimate std.error statistic   p.value
##   <chr>                <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)          23.0      0.899      25.6  3.63e-120
## 2 interval_4(7,13]     -1.09     1.05       -1.04 2.98e-  1
## 3 interval_4(13,19]    -2.80     0.991      -2.82 4.83e-  3
## 4 interval_4(19,25]    -6.39     0.981      -6.51 1.02e- 10
```

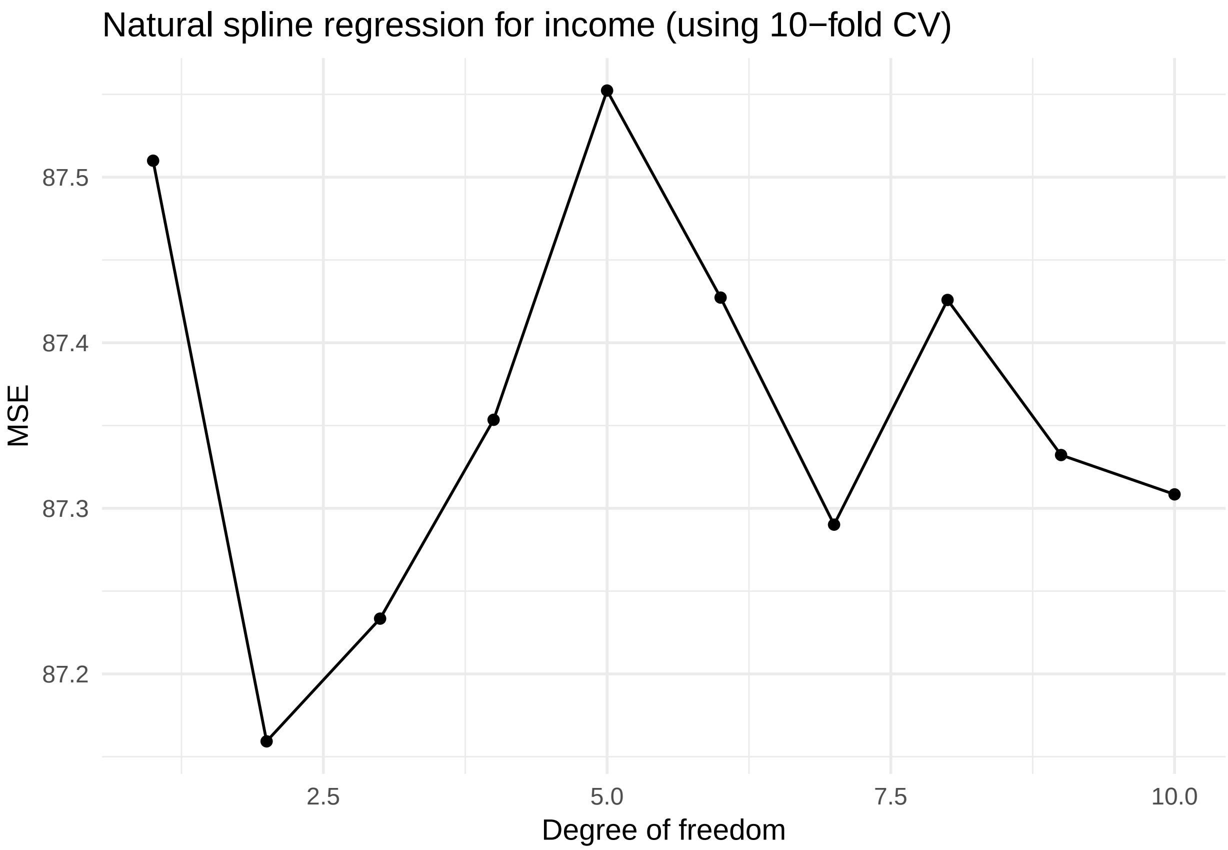
```
optimal_step_pred <- augment(optimal_step, newdata = inc_train) %>%
  mutate(pred_low = .fitted - 1.96 * .se.fit,
         pred_high = .fitted + 1.96 * .se.fit)

# plot the step function regression
ggplot(optimal_step_pred, aes(income06)) +
  geom_line(aes(y = .fitted)) +
  geom_line(aes(y = pred_low), linetype = 2) +
  geom_line(aes(y = pred_high), linetype = 2) +
  labs(title = "Step function regression",
       x = "Income level",
       y = "predicted egalitarianism")
```



3. Natural regression spline

```
nrs_mse <- vector(length = 10)

# calcualte the MSE for natural regression spline
for(i in 1:10){
  splited_data <- inc_train_cv$splits[[i]]
  train_data <- analysis(splited_data)
  holdout <- assessment(splited_data)
  actual <- holdout$egalit_scale
  for(k in 1:10){
    mod <- glm(data = train_data,
               egalit_scale ~ ns(income06, k))
    pred <- predict(mod, holdout)
    mse_tempo <- sum((pred - actual)**2) / length(pred)
    nrs_mse[k] <- nrs_mse[k] + mse_tempo
  }
}

# get the average MSE of natural regression spline
nrs_mse <- nrs_mse / 10
optimal_df <- which.min(nrs_mse)

# plot the MSE of natural regression spline
tibble(MSE = nrs_mse, df = 1:10) %>%
```

```r
  ggplot(aes(x= df, y = MSE)) + geom_point() + geom_line()+
  labs(title = "Natural spline regression for income (using 10-fold CV)",
         x = "Degree of freedom",
         y = "MSE")
```



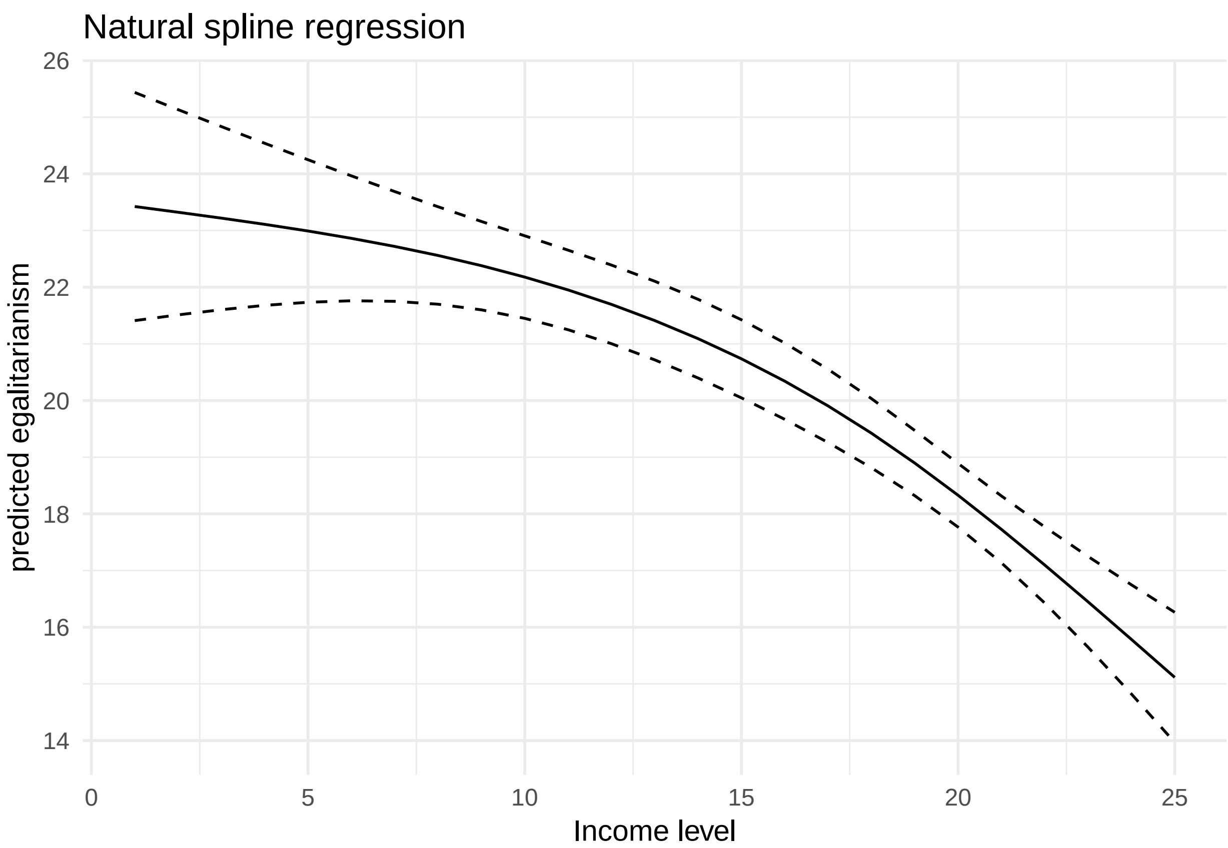
The optimal degree of freedom for a natural spline model is 2.

```r
# fit the optimal natural regression spline
optimal_nrs <- glm(data = inc_train,
               egalit_scale ~ ns(income06, optimal_df))

tidy(optimal_nrs)
```

```
## # A tibble: 3 x 5
##   term                        estimate std.error statistic  p.value
##   <chr>                          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                    23.4      1.03      22.8  6.76e-99
## 2 ns(income06, optimal_df)1    -7.74      2.10      -3.69 2.36e- 4
## 3 ns(income06, optimal_df)2    -7.53      0.813     -9.27 6.53e-20
```

```r
optimal_nrs_pred <- augment(optimal_nrs, newdata = inc_train) %>%
  mutate(pred_low = .fitted - 1.96 * .se.fit,
         pred_high = .fitted + 1.96 * .se.fit)

# plot the optimal natural regression spline model
ggplot(optimal_nrs_pred, aes(income06)) +
  geom_line(aes(y = .fitted)) +
  geom_line(aes(y = pred_low), linetype = 2) +
```

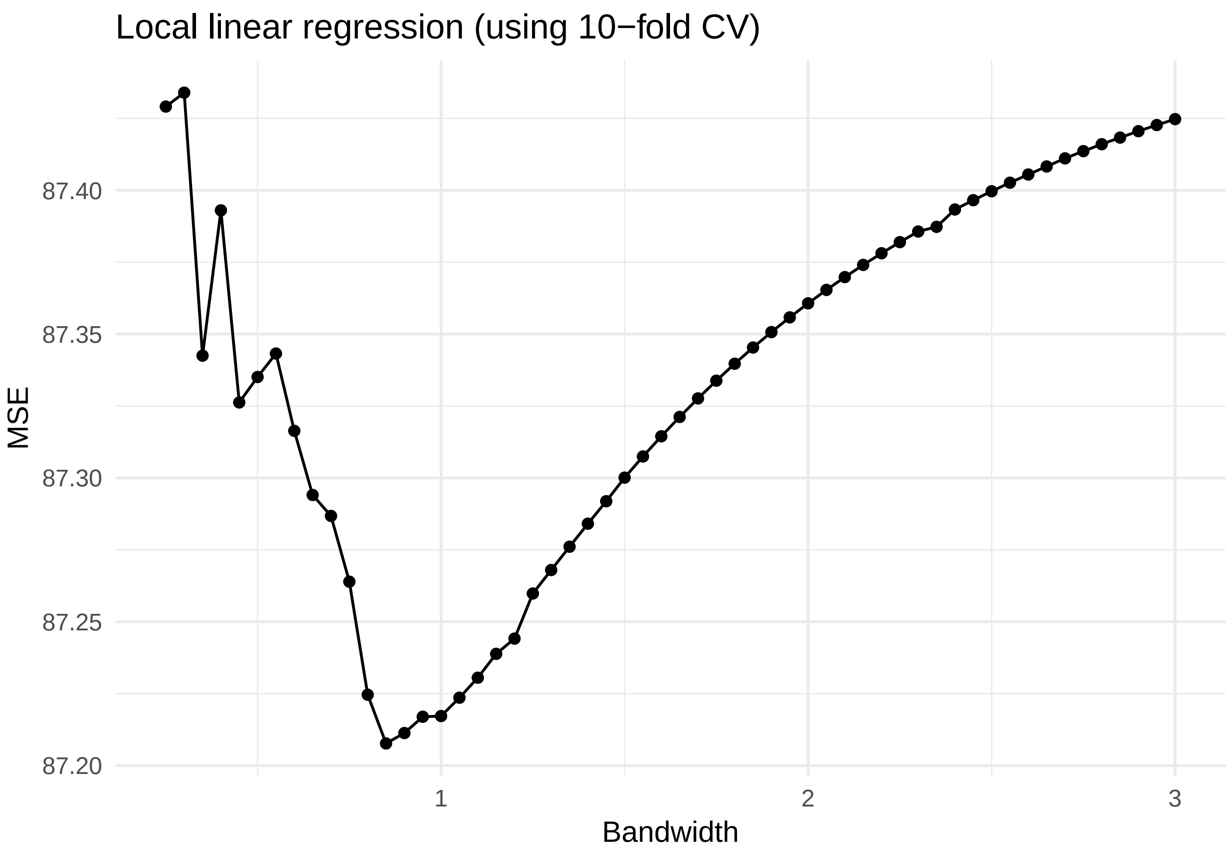
```r
  geom_line(aes(y = pred_high), linetype = 2) +
  labs(title = "Natural spline regression",
       x = "Income level",
       y = "predicted egalitarianism")
```



4. Local linear regression model

```r
span_range <- seq(from = 0.25, to = 3, by = 0.05)
local_mse <- vector(length = length(span_range))

# calcualte the MSE of local linear regression model
for (i in 1:10){
  splited_data <- inc_train_cv$splits[[i]]
  train_data <- analysis(splited_data)
  holdout <- assessment(splited_data)
  actual <- holdout$egalit_scale
  for (s in 1:length(span_range)){
    mod <- loess(data = train_data, egalit_scale ~ income06,
                 degree = 1, span = span_range[s])
    pred <- predict(mod, holdout)
    mse_tempo <- sum((pred - actual)**2) / length(pred)
    local_mse[s] <- local_mse[s] + mse_tempo
  }
}
```

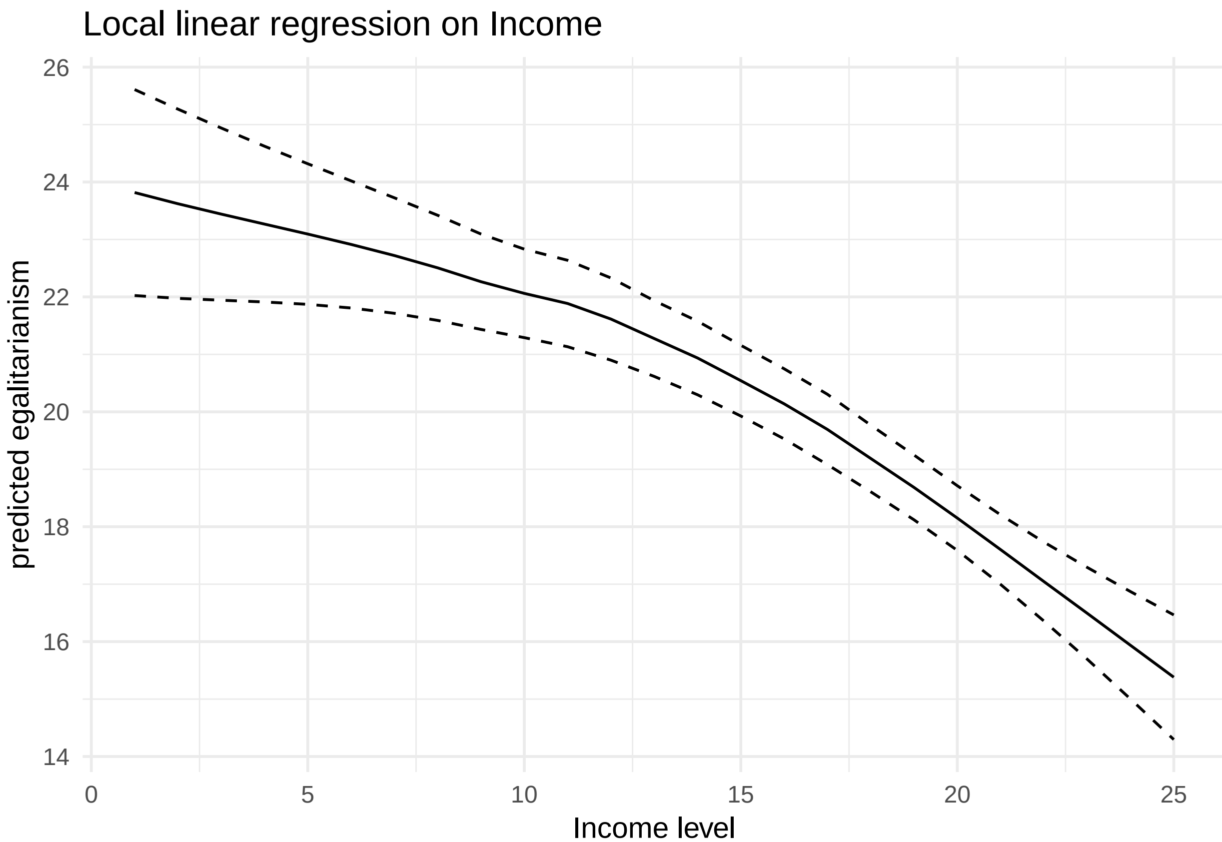
```r
# get the average MSE of local linear regression model
local_mse <- local_mse / 10
opt_span_index <- which.min(local_mse)
opt_span <- span_range[opt_span_index]

# plot the MSE of local linear regression model
tibble(MSE = local_mse, span = span_range) %>%
  ggplot(aes(x= span_range, y = MSE)) + geom_point() + geom_line() +
  labs(title = "Local linear regression (using 10-fold CV)",
       x = "Bandwidth",
       y = "MSE")
```



The optimal span is 0.85.

```r
# fit the optimal local linear regression model
optimal_local <- loess(data = inc_train,
                 egalit_scale ~ income06,
                 degree = 1, span = opt_span)

optimal_local_pred <- augment(optimal_local, newdata = inc_train) %>%
  mutate(pred_low = .fitted - 1.96 * .se.fit,
         pred_high = .fitted + 1.96 * .se.fit)

# plot the optimal local linear regression model
ggplot(optimal_local_pred, aes(income06)) +
  geom_line(aes(y = .fitted)) +
  geom_line(aes(y = pred_low), linetype = 2) +
  geom_line(aes(y = pred_high), linetype = 2) +
```

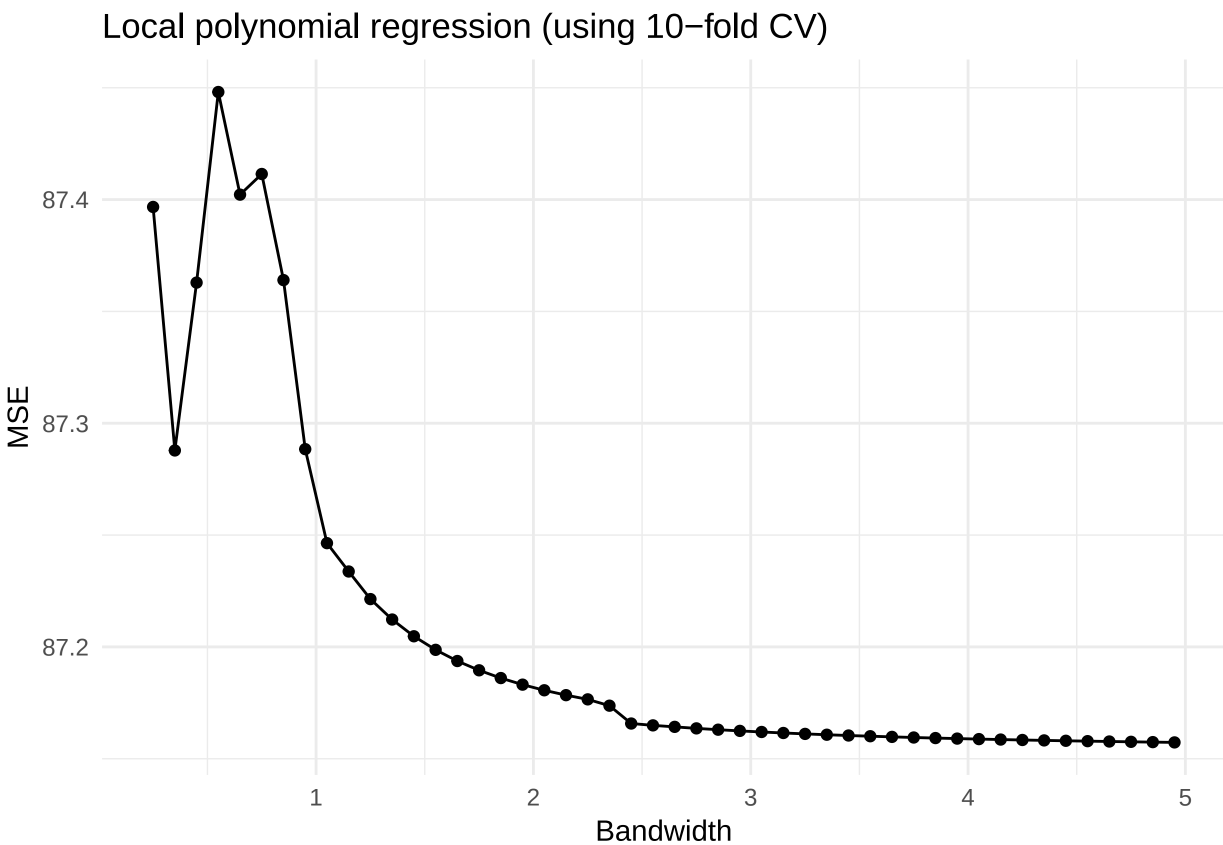
```
  labs(title = "Local linear regression on Income",
       x = "Income level",
       y = "predicted egalitarianism")
```



5. Local polynomial regression model

```
span_range <- seq(from = 0.25, to = 5, by = 0.1)
local_poly_mse <- vector(length = length(span_range))

# calcualte the MSE of local polynomial regression model
for(i in 1:10){
  splited_data <- inc_train_cv$splits[[i]]
  train_data <- analysis(splited_data)
  holdout <- assessment(splited_data)
  actual <- holdout$egalit_scale
  for(s in 1:length(span_range)){
    mod <- loess(data = train_data, egalit_scale ~ income06,
                 degree = 2, span = span_range[s])
    pred <- predict(mod, holdout)
    mse_tempo <- sum((pred - actual)**2) / length(pred)
    local_poly_mse[s] <- local_poly_mse[s] + mse_tempo
  }
}

# get the average MSE of local polynomial regression model
```

```r
local_poly_mse <- local_poly_mse / 10
opt_span_index <- which.min(local_poly_mse)
opt_span <- span_range[opt_span_index]

# plot the MSE of local polynomial regression model
tibble(MSE = local_poly_mse, span = span_range) %>%
  ggplot(aes(x= span_range, y = MSE)) + geom_point() + geom_line() +
  labs(title = "Local polynomial regression (using 10-fold CV)",
       x = "Bandwidth",
       y = "MSE")
```



The MSE decreases as the bandwidth increases. Therefore, I would use 5 as the optimal bandwidth to fit the local polynomial regression model.

```r
# fit the optimal local polynomial regression model
optimal_local_poly <- loess(data = inc_train, egalit_scale ~ income06,
                            degree = 2, span = 5)

optimal_local_poly_pred <- augment(optimal_local_poly, newdata = inc_train) %>%
  mutate(pred_low = .fitted - 1.96 * .se.fit,
         pred_high = .fitted + 1.96 * .se.fit)

# plot the optimal local polynomial regression model
ggplot(optimal_local_poly_pred, aes(income06)) +
  geom_line(aes(y = .fitted)) +
  geom_line(aes(y = pred_low), linetype = 2) +
  geom_line(aes(y = pred_high), linetype = 2) +
```

```
labs(title = "Local polynomial regression on Income",
     x = "Income level",
     y = "predicted egalitarianism")
```